document.addEventListener('DOMContentLoaded', () => {

  const navbarContainer1 = document.getElementById('navbarContainer1');

  // Fetch and include the navbar HTML

  fetch('navbar.html')

    .then(response => response.text())

    .then(html => {

      navbarContainer1.innerHTML = html;

      // Add event listener for the logout button

      const logoutBtn = document.getElementById('logoutBtn');

      logoutBtn.addEventListener('click', logout);

      // Retrieve the user's name from localStorage

      const userName = localStorage.getItem('userName');

      // Display the user's name on the page

      const welcomeUser = document.getElementById('welcomeUser');

      welcomeUser.textContent = `Welcome ${userName}`;

    })

    .catch(error => {

      console.error('Error fetching navbar:', error);

    });

  // Logout function

  function logout() {

    window.location.href = 'index.html';

  }

});

<li><button id="logoutBtn">Logout</button></li>

View all note / js

document.addEventListener('DOMContentLoaded', () => {

    const tableBody = document.querySelector('#view\_allNotes tbody');

    // Fetch all notes from the server

    fetch('http://127.0.0.1:9002/viewallnotes')

      .then(response => response.json())

      .then(notes => {

        // Generate HTML for each note and append it to the table body

        let serialNumber = 1; // Initialize the serial number

        notes.forEach(note => {

          const row = document.createElement('tr');

          row.innerHTML = `

            <td>${serialNumber}</td>

            <td>${note.uploadedBy}</td>

            <td>${new Date(note.uploadingDate).toLocaleDateString()}</td>

            <td>${note.branch}</td>

            <td>${note.subject}</td>

            <td><a href="http://127.0.0.1:9002/viewallnotes/${note.\_id}/download" target="\_blank">Download</a></td>

            <td>${note.fileType}</td>

            <td>${note.description}</td>

            <td>${note.status}</td>

          `;

          tableBody.appendChild(row);

          serialNumber++; // Increment the serial number for the next note

        });

      })

      .catch(error => {

        console.error(error);

        // Handle error if fetching notes fails

      });

  });

View all note/server

const express = require('express');

const router = express.Router();

const Note = require('../models/Note');

const path = require('path');

// GET all notes

router.get('/', async (req, res) => {

  try {

    const notes = await Note.find();

    res.json(notes);

  } catch (error) {

    console.error(error);

    res.status(500).json({ message: 'Server error' });

  }

});

// Download a note

router.get('/:id/download', async (req, res) => {

  try {

    const note = await Note.findById(req.params.id);

    if (!note) {

      return res.status(404).json({ message: 'Note not found' });

    }

    const filePath = path.join(\_\_dirname, '../uploads', note.file);

    res.download(filePath);

  } catch (error) {

    console.error(error);

    res.status(500).json({ message: 'Server error' });

  }

});

module.exports = router;

<td>

          <button class="edit-btn" onclick="editNote('${note.\_id}')">Edit</button>

          <button class="delete-btn" onclick="deleteNote('${note.\_id}')">Delete</button>

        </td>

function editNote(noteId) {

  // Implement your edit note logic here

  console.log(`Editing note with ID: ${noteId}`);

}

function deleteNote(noteId) {

  // Implement your delete note logic here

  console.log(`Deleting note with ID: ${noteId}`);

}

Upload notes/js with user id

document.addEventListener("DOMContentLoaded", () => {

  // Get the form and submit button

  const form = document.querySelector(".form");

  const submitBtn = document.querySelector(".form-btn");

  // Add submit event listener to the form

  form.addEventListener("submit", uploadNotes);

  // Function to handle form submission

  function uploadNotes(event) {

    event.preventDefault();

    // Retrieve the userId and authentication token from local storage

    const userId = localStorage.getItem("userId");

    const token = localStorage.getItem("authToken");

    const branch = document.querySelector("select[name='branch']").value;

    const subject = document.querySelector("input[name='subject']").value;

    const fileType = document.querySelector("select[name='file-type']").value;

    const notesFile = document.querySelector("input[name='notes-file']").files[0];

    const description = document.querySelector("#description").value;

    // Perform validation

    if (branch === "select" || subject === "" || fileType === "select" || !notesFile) {

      alert("Please fill in all the required fields.");

      return;

    }

    // Prepare form data

    const formData = new FormData();

    formData.append("userId", userId); // Include userId in the form data

    formData.append("branch", branch);

    formData.append("subject", subject);

    formData.append("fileType", fileType);

    formData.append("notes-file", notesFile);

    formData.append("description", description);

    // Send the form data to the server using AJAX

    const xhr = new XMLHttpRequest();

    xhr.open("POST", "http://localhost:9002/upload");

    // Set the authentication token in the request headers

    xhr.setRequestHeader("Authorization", `Bearer ${token}`);

    xhr.onreadystatechange = function () {

      if (xhr.readyState === XMLHttpRequest.DONE) {

        if (xhr.status === 201) {

          // Successful response from the server

          alert("Notes uploaded successfully!");

          window.location.href = "user\_home.html";

          form.reset();

        } else {

          // Error response from the server

          alert("Error uploading notes. Please try again.");

        }

      }

    };

    xhr.send(formData);

  }

});

Upload note/server userid

const express = require('express');

const multer = require('multer');

const router = express.Router();

const Note = require('../models/Note');

// Middleware to authenticate the user

function authenticateUser(req, res, next) {

  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith('Bearer ')) {

    return res.status(401).json({ message: 'Unauthorized' });

  }

  const token = authHeader.split(' ')[1]; // Extract the token from the "Bearer" scheme

  // Here, you should implement the logic to verify the token and retrieve the userId from it

  // For example, using a JWT verification library or any other authentication mechanism

  // Assuming you have already verified the userId and saved it in local storage during login

  const userId = localStorage.getItem('userId'); // Retrieve the verified userId from local storage

  if (!userId) {

    return res.status(401).json({ message: 'Unauthorized' });

  }

  req.userId = userId; // Attach the userId to the request for future use

  next();

}

// Configure multer storage

const storage = multer.diskStorage({

  destination: function (req, file, cb) {

    cb(null, 'uploads'); // Specify the destination folder to store the uploaded files

  },

  filename: function (req, file, cb) {

    // Generate a unique filename for the uploaded file

    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1e9);

    const fileExtension = file.originalname.split('.').pop();

    cb(null, file.fieldname + '-' + uniqueSuffix + '.' + fileExtension);

  }

});

// Create multer instance with the configured storage

const upload = multer({ storage });

// Upload notes - Add the authenticateUser middleware as the first parameter

router.post('/', authenticateUser, upload.single('notes-file'), async (req, res) => {

  try {

    const { branch, subject, fileType, description } = req.body;

    // Access the uploaded file details using req.file

    const file = req.file;

    if (!file) {

      return res.status(400).json({ message: 'No file uploaded' });

    }

    const newNote = new Note({

      userId: req.userId, // Use the authenticated userId

      branch,

      subject,

      file: file.filename, // Store the filename in the database

      fileType,

      description,

    });

    await newNote.save();

    res.status(201).json({ message: 'Note uploaded successfully' });

  } catch (error) {

    console.error(error);

    res.status(500).json({ message: 'Server error' });

  }

});

module.exports = router;

import { GiNotebook } from "react-icons/gi";

import {ImCancelCircle} from "react-icons/im";

import {FiBookOpen} from "react-icons/fi";

import {TbFileCheck} from "react-icons/tb";

import { BiUser } from "react-icons/bi";

import { FiLogIn } from "react-icons/fi";

import { TbFileUpload, TbFileDownload } from "react-icons/tb";

import { FaFacebook, FaTwitter, FaInstagram, FaDribbble} from "react-icons/fa";

import {FaUpload}from "react-icons/fa";

noteSchema.index({ userId: 1, branch: 1 }); // Index userId and branch fields